What is claimed is:

1 A distributed debugger system, which debugs a distributed system which is configured by a program run on plural computers, comprising:

a program manager for executing a management from a predetermined computer to other computers via a network interconnecting said plural computers, said management being related to the setting status and execution status of a debug object program executed on each of said plural computers.

2 The distributed debugger system defined in Claim 1, wherein said program manager comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including a setting-status manager for managing the setting of each of debuggers constructing said distributed debugger system and communication means for communicating with other controllers or said executors;

each of said executors including a setting-status manager for managing the setting of each of debuggers constructing said distributed debugger system and communication means for communicating with other

controllers or executors;

said setting-status manager changing its setting content when a change in setting is instructed and then notifying a setting-status manager in other controller or said executor of the changed content using said communication means, while said setting-status manager changes its setting content in response to notification;

whereby said distributed system to be debugged is debugged using the same setting on all computers in which said distributed debugger operates.

3 The distributed debugger system defined in Claim 2, wherein said program manger comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including an execution-status manager for managing an execution status of each of debuggers constructing said distributed debugger system, and communication means for communicating with other controllers or executors;

each of said executors including said execution-status manager for managing the execution status of said debugger, a process manager for managing a debug

object program, and communication means for communicating with other controllers or executors;

wherein said execution-status manager changes its setting content when being instructed to change the execution status; said communication means notifies said execution-status manager in other controller or execution-status manager of the changed content; and said execution-status manager changes its status in response to the notification and instructs said process manager to instruct a change of operation; whereby the same execution status is maintained on all computers on which said distributed debugger system operates.

4 The distributed debugger defined in Claim 2, wherein said program manager comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including a user interface for interpreting a request for displaying the status of a debug object program from a user and displaying the results; place decision means for specifying one or more existence places in the status specified based on the nature of the status when the status of a

debug object program is specified and based on the execution status of a debug object program acquired from said process manager, and communication means for communicating with other controller or executor;

each of executors including a process manager for managing a debug object program and communication means for communicating with other controller or executor;

said user interface inquiring said place decision means in response to a request for displaying the status of a debug object program from a user and then acquiring one or more existence places of the status and transmitting a status capture request to said process managers at all existence places via said communication means;

said process manager checking the execution status of a debug object program under management in response to a status capture request and transmitting results to said user interface at the request source;

said user interface receiving one or more results and then outputting said results in a batch mode to said user, whereby said user acquires the status of a distributed system to be objected, without recognizing its existence place.

5 The distributed debugger system defined in Claim 2,

wherein said program manager and said debug object manager are realized on the same distributed system construction foundation and wherein program manager uses a communication function provided by said distributed system construction foundation.

6 The distributed debugger system defined in Claim 2, wherein said program manager comprises:

said setting-status manager for holding as a setting status a debug object program activation method;

a remote debugger activator for activating said executor on a remote computer; and

a user interface for interpreting a debug object program from a user and a specification of an execution start place thereof;

said user interface receiving a specification of said debug object program from said user and then notifying said setting-status manager of the specification;

said user interface receiving the specification of said execution start place of said debug object program from said user and instructing said executor specified by said remote debugger activator to activate said executor;

said remote debugger activator activating said executor on a specified computer;

said user interface instructs said activated executor to start the execution of said debug object program;

said process manager in said executor instructed acquiring information about a debug object program held in said setting-status manager to start execution of said debug object program;

whereby execution of a debug object program is started on a computer different from the computer operated by said user.

7 The distributed debugger system defined in Claim 1, wherein said program manger comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including an execution-status manager for managing an execution status of each of debuggers constructing said distributed debugger system, and communication means for communicating with other controllers or executors;

each of said executors including said execution-status manager for managing the execution status of said debugger, a process manager for managing a debug object program, and communication means for communicating with other controllers or executors;

wherein said execution-status manager changes its setting content when being instructed to change the execution status; said communication means notifies said execution-status manager in other controller or execution-status manager of the changed content; and said execution-status manager changes its status in response to the notification and instructs said process manager to instruct a change of operation; whereby the same execution status is maintained on all computers on which said distributed debugger system operates.

8 The distributed debugger defined in Claim 7, wherein said program manager comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including a user interface for interpreting a request for displaying the status of a debug object program from a user and displaying the results; place decision means for specifying one or more existence places in the status specified based on the nature of the status when the status of a debug object program is specified and based on the execution status of a debug object program acquired

from said process manager, and communication means for communicating with other controller or executor;

each of executors including a process manager for managing a debug object program and communication means for communicating with other controller or executor;

said user interface inquiring said place decision means in response to a request for displaying the status of a debug object program from a user and then acquiring one or more existence places of the status and transmitting a status capture request to said process managers at all existence places via said communication means;

said process manager checking the execution status of a debug object program under management in response to a status capture request and transmitting results to said user interface at the request source;

said user interface receiving one or more results and then outputting said results in a batch mode to said user, whereby said user acquires the status of a distributed system to be objected, without recognizing its existence place.

9 The distributed debugger system defined in Claim 8, wherein said program manager and said debug object manager are realized on the same distributed system construction

foundation and wherein program manager uses a communication function provided by said distributed system construction foundation.

10 The distributed debugger system defined in Claim 7, wherein said program manager interprets a request for changing the status of a debug object program, from a user, and instructs said execution-status manager to change the status.

11 The distributed debugger system defined in Claim 10, wherein said program manager and said debug object manager are realized on the same distributed system construction foundation and wherein program manager uses a communication function provided by said distributed system construction foundation.

12 The distributed debugger system defined in Claim 7, wherein said process manager in said executor within said program manager changes its operation according to the operation of a debug object program and changes the status of said execution-status manager within the same executor based on the changed content.

13 The distributed debugger system defined in Claim 12, wherein said program manager and said debug object manager are realized on the same distributed system construction foundation and wherein program manager uses a communication function provided by said distributed system

construction foundation.

14 The distributed debugger system defined in Claim 7, wherein said program manager and said debug object manager are realized on the same distributed system construction foundation and wherein program manager uses a communication function provided by said distributed system construction foundation.

15 The distributed debugger defined in Claim 1, wherein said program manager comprises:

at least one of controllers for receiving instructions from a user;

at least one of executors connected to said debug object program;

each of said controllers including a user interface for interpreting a request for displaying the status of a debug object program from a user and displaying the results; place decision means for specifying one or more existence places in the status specified based on the nature of the status when the status of a debug object program is specified and based on the execution status of a debug object program acquired from said process manager, and communication means for communicating with other controller or executor;

each of executors including a process manager for managing a debug object program and communication

means for communicating with other controller or executor;

said user interface inquiring said place decision means in response to a request for displaying the status of a debug object program from a user and then acquiring one or more existence places of the status and transmitting a status capture request to said process managers at all existence places via said communication means;

said process manager checking the execution status of a debug object program under management in response to a status capture request and transmitting results to said user interface at the request source;

said user interface receiving one or more results and then outputting said results in a batch mode to said user, whereby said user acquires the status of a distributed system to be objected, without recognizing its existence place.

16 The distributed debugger system defined in Claim 1, wherein said program manager and said debug object manager are realized on the same distributed system construction foundation and wherein program manager uses a communication function provided by said distributed system construction foundation.

17 A debugging method, suitable in use for a distributed

debugger system that debugs a distributed system configured of a program which runs on plural computers, said method comprising the step of:

implementing a program management from a predetermined computer to other computers via a network interconnecting said computers, said program management being related to the setting status and execution status of a debug object program to be executed on each of said computers.

18 The debugging method defined in Claim 17, wherein said program management step comprises the sub-steps of:

transmitting an instruction of a setting change from a user or other computers to each of said debuggers constructing said distributed debugger system;

changing the setting in response to said instruction;

deciding whether or not said instruction has come from other computers; and

notifying said debuggers on said other computers of said instruction via communications when said instruction has not come via communications;

whereby a distributed debugger system to be debugged is debugged using the same settings on all computers on which said distributed debugger system operates.

19 The debugging method defined in Claim 18, wherein each of debuggers constructing said distributed debugger system

includes the steps of:

receiving an instruction of a change in execution status;

deciding whether or not said instructed status corresponds to a change to the same status as a current status;

changing the status when the instructed status is not the same as the current status;

deciding whether or not said execution status change is instructed according to a change of the management status of a debug object program;

instructing said debugger on other computer to change the execution status via communications when the status change is instructed due to a change of the management status of said debug object program;

deciding whether or not a debugger is said debugger which is managing said debug object program when the status change is not instructed due to a change of the management status of said debug object program; and

changing the management status of said debug object program in accordance with the status changed when a debugger is said debugger which is managing said debug object program;

whereby the same execution status is maintained on all

computers on which said distributed debugger system operates.

20 The debugging method defined in Claim 18, wherein each of said debuggers constructing said distributed debugger system implements the steps of:

receiving an instruction for displaying a status from a user;

interpreting said instruction;

specifying one or more specified status existence places;

transmitting a status capture request to debuggers at all existence places;

acquiring the status of a debug object program by means of said debugger which has received said status capture request;

acknowledging the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a request source; and

outputting contents of received replies in a batch mode to said user;

whereby said user acquires the status inside said distributed system to be debugged, without recognizing the existence place.

21 The debugging method defined in Claim 17, wherein each

of debuggers constructing said distributed debugger system includes the steps of:

receiving an instruction of a change in execution status;

deciding whether or not said instructed status corresponds to a change to the same status as a current status;

changing the status when the instructed status is not the same as the current status;

deciding whether or not said execution status change is instructed according to a change of the management status of a debug object program;

instructing said debugger on other computer to change the execution status via communications when the status change is instructed due to a change of the management status of said debug object program;

deciding whether or not a debugger is said debugger which is managing said debug object program when the status change is not instructed due to a change of the management status of said debug object program; and

changing the management status of said debug object program in accordance with the status changed when a debugger is said debugger which is managing said debug object program;

whereby the same execution status is maintained on all computers on which said distributed debugger system operates.

22 The debugging method defined in Claim 21, wherein each of said debuggers constructing said distributed debugger system implements the steps of:

receiving an instruction for displaying a status from a user;

interpreting said instruction;

specifying one or more specified status existence places;

transmitting a status capture request to debuggers at all existence places;

acquiring the status of a debug object program by means of said debugger which has received said status capture request;

acknowledging the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a request source; and

outputting contents of received replies in a batch mode to said user;

whereby said user acquires the status inside said distributed system to be debugged, without recognizing the existence place.

23 The debugging method defined in Claim 21, wherein each of debuggers constructing said distributed debugger system, comprises the steps of:

receiving an instruction of a status change from a user; and

interpreting said instruction and then changing the execution status.

24 The debugging method defined in Claim 21, wherein each of debuggers constructing said distributed debugger system, comprises the steps of:

monitoring an operation status of a debug object program; changing the management status of said debug object program when specific requirements are satisfied during monitoring; and

instructing an execution status change in accordance with a management status change.

25 The debugging method defined in Claims 17, wherein each of said debuggers constructing said distributed debugger system implements the steps of:

receiving an instruction for displaying a status from a user;

interpreting said instruction;

specifying one or more specified status existence places;

transmitting a status capture request to debuggers at

all existence places;

acquiring the status of a debug object program by means of said debugger which has received said status capture request;

acknowledging the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a request source; and

outputting contents of received replies in a batch mode to said user;

whereby said user acquires the status inside said distributed system to be debugged, without recognizing the existence place.

26 A recording medium, on which a control program for a distributed debugger system for debugging a distributed system constructed by a program which runs on plural computers is recorded, said recording medium recording a program management step of executing a management from a specific computer to other computers via a network interconnecting plural computers, said management being related to the setting status and execution status of a debug object program executed on each of said computers.

27 The recording medium defined in Claim 26, wherein said program management step comprises the steps of:

receiving an instruction of a change of setting from a

user or other computers by means of each of debugger constructing said distributed debugger system;

changing the setting in response to said instruction;

deciding whether or not said instruction has come from other computers; and

notifying a debugger on other computers of the instruction via communications when said instruction is not received via communications;

whereby a distributed debugger to be debugged is debugged using the same setting on all computers on which said distributed debugger system operates.

28 The recording medium defined in Claim 27, wherein each of said debuggers constructing said distributed debugger system implements the steps of:

receiving an instruction of an execution status change;

deciding whether or not the instructed status is a change to the same status as a current status;

changing the status when the instructed status is not the same as the current status;

instructing an execution status change to said debugger on other computers via communications when a status change is instructed in accordance with a change of the management status of the debug object program;

deciding whether or not said debugger managing the debug object program when a status change is not

instructed in accordance with a change of the management status of the debug object program; and

changing the management status of a debug object program in accordance with the status changed when a debugger is said debugger managing the debug object program;

whereby the same execution status is maintained on all computers on which said distributed debugger system is operated.

29 The recording medium defined in Claim 27, wherein each of said debuggers constructing said distributed system, implements the steps of:

receiving an instruction for displaying the status from a user;

interpreting said instruction;

specifying one or more existence places in the specified status;

transmitting a status capture request to debuggers at all specified existence places;

acquiring the status of the debug object program by means of said debugger which has received the status capture request;

returning the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a

request source; and

outputting contents of said received replies to said user in a batch mode;

whereby said user acquires the status inside a distributed system to be debugged, without recognizing the existence place.

30 The recording medium defined in Claim 26, wherein each of said debuggers constructing said distributed debugger system implements the steps of:

receiving an instruction of an execution status change;

deciding whether or not the instructed status is a change to the same status as a current status;

changing the status when the instructed status is not the same as the current status;

instructing an execution status change to said debugger on other computers via communications when a status change is instructed in accordance with a change of the management status of the debug object program;

deciding whether or not said debugger managing the debug object program when a status change is not instructed in accordance with a change of the management status of the debug object program; and

changing the management status of a debug object program in accordance with the status changed when a debugger is said debugger managing the debug object

program;

whereby the same execution status is maintained on all computers on which said distributed debugger system is operated.

31 The recording medium defined in any one of Claim 30, wherein each of said debuggers constructing said distributed system, implements the steps of:

receiving an instruction for displaying the status from a user;

interpreting said instruction;

specifying one or more existence places in the specified status;

transmitting a status capture request to debuggers at all specified existence places;

acquiring the status of the debug object program by means of said debugger which has received the status capture request;

returning the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a request source; and

outputting contents of said received replies to said user in a batch mode;

whereby said user acquires the status inside a distributed system to be debugged, without

recognizing the existence place.

32 The recording medium defined in Claim 30, wherein each of said debuggers constructing said distributed system, implements the steps of:

receiving an instruction of a stat[redacted]ange from a user; and

interpreting the instruction and changing the execution status thereof.

33 The recording medium defined in Claim 30, wherein each of said debuggers constructing said distributed system, implements the steps of:

monitoring the operation status of a debug object program;

changing the management status of the debug object program when specific requirements are satisfied during monitoring; and

instructing a change in execution status in accordance with a change in management status.

34 The recording medium defined in Claim 26, wherein each of said debuggers constructing said distributed system, implements the steps of:

receiving an instruction for displaying the status from a user;

interpreting said instruction;

specifying one or more existence places in the

specified status;

transmitting a status capture request to debuggers at all specified existence places;

acquiring the status of the debug object program by means of said debugger which has received the status capture request;

returning the acquired status to said debugger being a request source;

receiving all replies by means of said debugger being a request source; and

outputting contents of said received replies to said user in a batch mode;

whereby said user acquires the status inside a distributed system to be debugged, without recognizing the existence place.